Habitly (Habit Tracker) — Viva/Cross-Question Cheat Sheet

Concise, judge-ready answers with extra likely cross-questions. Print this page to PDF.

How did you build the habit tracker?

- **Core platform**: Web app (mobile-first) using HTML, CSS, JavaScript.
- **Framework/libs**: Optional React; localStorage/IndexedDB for offline; optional Node/Express if backend is needed.
- **State/data**: Array of habit objects: `{ id, name, frequency, days[], streak, lastDone, reminders[] }`.
- **UI**: Calendar grid, daily checklist, streak counters, progress ring, simple charts.
- **Features**: Create/edit habits, daily check-in, streaks, reminders, basic analytics.

What did you use to build it?

- **Frontend**: HTML5, CSS3 (flex/grid), JavaScript (ES6).
- **Storage**:
    - No-backend: `localStorage` (simple) or `IndexedDB` (larger data).
    - Optional backend: Node.js + Express + JSON DB or MongoDB for sync.
- **Notifications**: Browser Notification API; if backend, cron or push.
- **Charts**: Custom SVG/CSS or tiny chart lib.
- **Testing**: Manual test cases; optional Jest for logic.

Why use these choices?

- **Beginner-friendly**: Matches team's HTML/CSS/JS skills.
- **Fast to build**: Local storage avoids backend complexity.
- **Offline-first**: Works without internet.
- **Clarity**: Logic is easy to explain to judges.

What is your habit tracker (project) about?

- **Goal**: Build consistent habits via simple daily check-ins and feedback.
- **Principles**: Streak motivation, small wins, reminders, visual progress.
- **Scope**: Personal habits (study, steps, water, meditation, etc.).

Why should someone use your tracker when many exist?

- **Simplicity**: Low friction — add habit, tap check-in, see streak.
- **Privacy**: Data stays on device by default.
- **Offline**: Works without connectivity; fast and ad-free.
- **Customization**: Flexible frequencies, multiple reminders, skip rules.
- **Lightweight**: Minimal permissions and fast loads.

On what basis does it work?

- **Behavioral**: Cue → Routine → Reward; streaks reinforce consistency; reminders act as cues; small wins drive adherence.
- **Data**: Daily logs → adherence metrics and streaks → feedback loop.

What logic does your tracker use?

- **Streak**: If done today and yesterday (or allowed gap), `streak++` ; else reset to 1. Weekly/monthly adjust by period.
- **Adherence**: `adherence = (completed / scheduled) * 100` .
- **Chain**: Show continuous marked days to avoid breaking chain.
- **Reminders**: Store times; compare periodically; fire notifications.
- **Edge cases**: Normalize to local midnight; handle timezone; strict mode for backfill.

How do you capture data?

- **User input**: Daily toggle; optional numeric value; notes.
- **Storage**: Per-day entries keyed by ISO date; update `lastDone` , `streak` , `history[date]` .

- **Permissions**: Only notifications (optional).
- **Sync**: Optional API for multi-device; otherwise local.

Extra likely cross-questions (with crisp answers)

What's your data model?

```
{ id, name, type: 'binary'|'count', targetPerPeriod, period: 'daily'|'weekly'|'monthly', history: { 'YYYY-MM-DD': { done, value } },
streak, longestStreak, reminders: ['HH:mm'], createdAt, updatedAt }
```

Missed days or backdating?

- Past days can be marked but tagged "late".
- Strict mode: late entries don't extend streaks.

What happens at midnight?

- Normalize date to local midnight; refresh today's schedule on next open.

Timezone changes?

- Store ISO date; compute "today" from device time each session.

Notifications?

- Request permission; compare times; trigger Notification API.
- Backend (optional): schedule pushes with cron.

Prevent duplicate notifications?

- Track last-fired timestamp; apply cooldown window.

Performance?

- O(1) access via date keys; render only visible days; memoize metrics.

Security & privacy?

- Local-first by default; if sync: token auth; minimal data.

Accessibility?

- High contrast, keyboard navigation, ARIA labels, large tap targets.

Why not Firebase?

- Mini project + offline-first → local storage is simpler; Firebase optional for sync.

Rest/skip days?

- Frequency and skip rules define expectations; adherence uses scheduled days only.

How did you test streak logic?

- Unit tests for date transitions, missed days, weekly habits; sample datasets.

Future improvements?

- Multi-device sync, social features, smart reminders, export data, AI suggestions.

How is this different from a to-do app?

- Focus on recurrence, streaks, adherence metrics, low-friction daily repetition.

App closed at reminder time?

- Fire on next open and show missed badge; backend push can deliver in background.

UI flow?

- Add habit → set frequency/reminders → daily check-in → view streaks/analytics.

Why no TypeScript?

- Small codebase + team skills → JavaScript keeps explanations simple.

Data export/import?

- JSON export/import for user control; easy migration later.

Avoid over-notifying?

- Daily per-habit limit; snooze; stop after completion.

Weekly/monthly habits?

- Track counts per period; streak increments per successfully completed period.

Short demo script

- Add “Drink Water”, daily, 3 reminders.
- Mark today done with numeric input; progress ring hits 100%.
- Show streak increment; view last 7 days adherence chart.
- Enable strict mode; backdate a day → streak doesn’t increase.

Why this design?

- Optimized for behavior change with minimal friction.
- Transparent, verifiable logic.
- Offline-first, privacy-respecting.
- Extensible to backend if needed.
